

Unidad 3

Arquitectura del conjunto de instrucciones.

Arquitectura de procesadores.

Interrupción y concurrencia.



Definiciones Instrucciones

Conjunto de Instrucciones: colección de todas las instrucciones que puede ejecutar un procesador se le denomina conjunto de instrucciones.

Arquitectura del conjunto de instrucciones: es la completa descripción de dicho conjunto de instrucciones.

- Determina cómo debe ser el hardware del procesador y cómo debe estar organizado. Por ello, los manuales de usuario de los procesadores se enfocan principalmente en la descripción del conjunto de instrucciones que el procesador puede ejecutar, así como de los elementos del procesador que son accesibles al programador.

Las instrucciones se definen y almacenan en la memoria del procesador en lenguaje binario, lo cual constituye el llamado código máquina. El lenguaje que sustituye los códigos de operación binarios y las direcciones por nombres simbólicos se denomina lenguaje ensamblador.

El formato de las instrucciones se representa mediante una caja rectangular simbolizando los bits de la instrucción en binario. Estos bits se dividen en grupos llamados campos:

- Campo de código de operación (opcode): que especifica la operación a realizar.
- Campo de dirección(es): que proporciona direcciones de memoria o de registros.
- Campo de modo: que especifica la forma en que se interpreta el campo de direcciones.

Opcode	Modo	Dirección u operando
--------	------	----------------------

CICLO DE OPERACIÓN BÁSICO DE UN PROCESADOR

La unidad de control ejecuta las instrucciones de un programa efectuando la siguiente secuencia de pasos:

- 1.- Obtener la instrucción de memoria. Almacenarla en un registro de control.**
- 2.- Decodificar la instrucción.**
- 3.- Localizar los operandos empleados en la instrucción.**
- 4.- Obtener de la memoria los operandos (si fuese necesario)**
- 5.- Ejecutar la operación en la ruta de datos.**
- 6.- Almacenar el resultado en un lugar adecuado.**
- 7.- Volver al paso 1 y para procesar la siguiente instrucción.**

La unidad de control del procesador consta de un registro especial, el contador de programa PC (Program Counter). Su contenido apunta a la posición de memoria de la instrucción que se va a ejecutar a continuación, y se incrementa cada vez que se lee una instrucción del programa almacenado en la memoria.

La decodificación del paso 2 determina la operación a ejecutar y el modo de direccionamiento de la instrucción.

En el paso 3, los operandos se localizan según el modo de direccionamiento y el campo de direcciones de la instrucción.

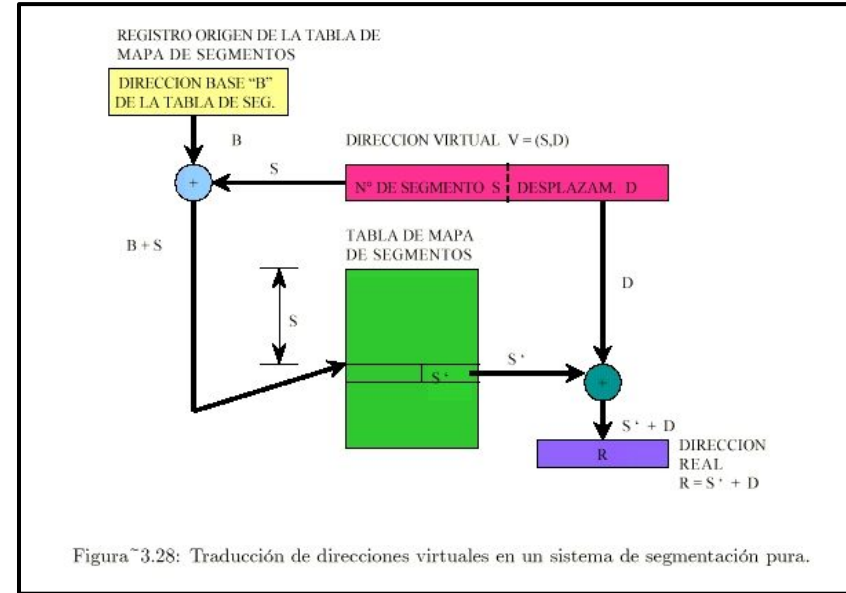
El procesador ejecuta la instrucción sobre los operandos, almacena el resultado y regresa al paso 1 para obtener la siguiente instrucción del programa.



Modos de direccionamiento

Los modos de direccionamiento son las diferentes maneras de especificar un operando dentro de una instrucción en lenguaje ensamblador.

Un modo de direccionamiento especifica la forma de calcular la dirección de memoria efectiva de un operando mediante el uso de la información contenida en registros y/o constantes, contenida dentro de una instrucción de la máquina o en otra parte.



Modos de direccionamiento

El modo de direccionamiento especifica una regla para interpretar o modificar el campo de direcciones de la instrucción antes de que se haga realmente referencia al operando. A la dirección del operando generada mediante la aplicación de esa regla se la denomina dirección efectiva.

Los procesadores utilizan técnicas de modo de direccionamiento para ajustarse a las siguientes características:

- Proporcionar flexibilidad al usuario en la programación mediante punteros a memoria, contadores para el control de bucles, indexar datos y reubicar programas.
- Reducir el número de bits de los campos de direcciones de la instrucción.

Disponer de varios modos de direccionamiento proporciona al programador experimentado la posibilidad de escribir programas que requieran pocas instrucciones. Sin embargo, el uso de modos de direccionamiento complejo puede implicar un mayor tiempo de ejecución.



Tipos de direccionamiento

Implícito:

En este modo de direccionamiento no es necesario poner ninguna dirección de forma explícita, ya que en el propio código de operación se conoce la dirección de el/los operando/s al (a los) que se desea acceder o con el/los que se quiere operar.

El operando se especifica implícitamente en el código de operación, por lo que la instrucción no necesita de un campo de direcciones.

Supongamos una arquitectura de pila, las operaciones aritméticas no requieren direccionamiento explícito por lo que se ponen como: - add - sub ...

Porque cuando se opera con dos datos en esta arquitectura se sabe que son los dos elementos del tope de la pila.

Ejemplos:

- De una pila: 1 2 3 4 5 6 <- pila top() es 1 ntop() es 2: Donde top() representa el tope de la pila y ntop() el siguiente al tope de la pila y son estos argumentos con los que se opera al llamar a una orden en concreto.
- INC A (incrementa el contenido de A en 1)

Tipos de direccionamiento

Inmediato

En la instrucción está incluido directamente el operando.

En este modo el operando es especificado en la instrucción misma. En otras palabras, una instrucción de modo inmediato tiene un campo de operando en vez de un campo de dirección. El campo del operando contiene el operando actual que se debe utilizar en conjunto con la operación especificada en la instrucción. Las instrucciones de modo inmediato son útiles para inicializar los registros en un valor constante.

Cuando el campo de dirección especifica un registro del procesador, la instrucción se dice que está en el modo de registro.

Su valor es fijo, por lo que se suele utilizar en operaciones aritméticas o para definir constantes y variables. Como ventaja, no se requiere acceso adicional a memoria para obtener el dato, pero el tamaño del operando está limitado por el tamaño del campo de direccionamiento.

Las desventajas principales son que el valor del dato es constante y el rango de valores que se pueden representar está limitado por el tamaño de este operando. Ejemplo: **MOV A,#17H**

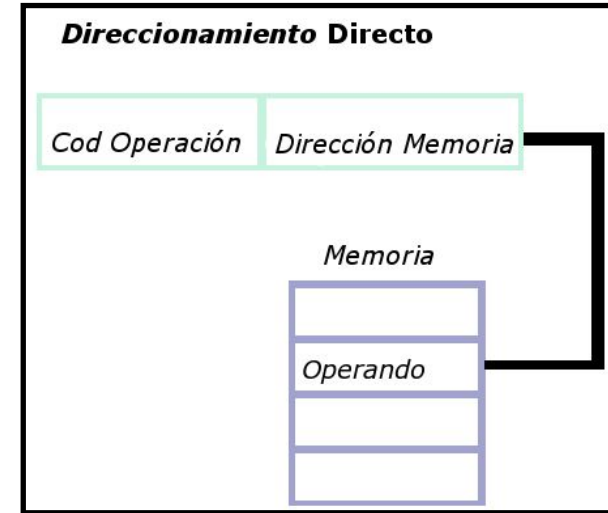


Tipos de direccionamiento

Directo

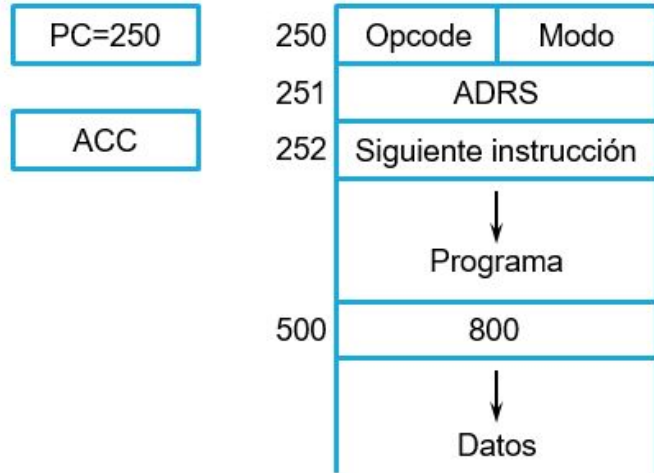
El campo de operando en la instrucción contiene la dirección en memoria donde se encuentra el operando.

En este modo la dirección efectiva es igual a la parte de dirección de la instrucción. El operando reside en la memoria y su dirección es dada directamente por el campo de dirección de la instrucción. En una instrucción de tipo ramificación el campo de dirección especifica la dirección de la rama actual. Si hace referencia a un registro de la máquina, el dato estará almacenado en este registro y hablaremos de direccionamiento directo a registro; si hace referencia a una posición de memoria, el dato estará almacenado en esta dirección de memoria (dirección efectiva) y hablaremos de direccionamiento directo a memoria. Estos modos de direccionamiento tienen una forma muy simple y no hay que hacer cálculos para obtener la dirección efectiva donde está el dato. El tamaño del operando, en el caso del direccionamiento directo a registro, dependerá del número de registros que tenga la máquina; en el direccionamiento directo a memoria, dependerá del tamaño de la memoria. Ejemplo: **MOV A,17H**



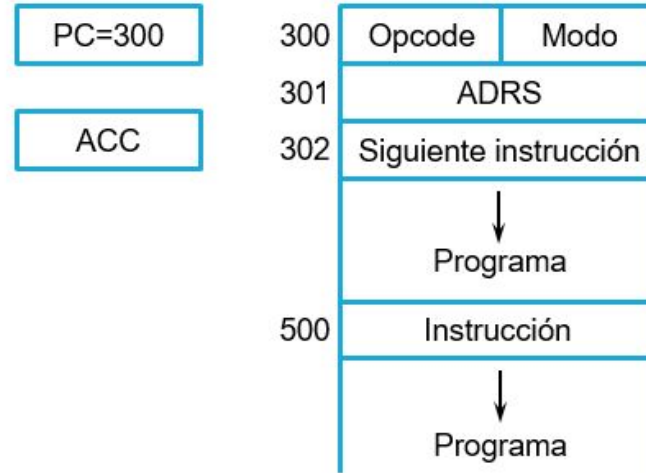
Tipos de direccionamiento

(a) Transferencia de datos



Opcode: Carga ACC
Modo: Dirección directa
ADRS: 500
Operación: $ACC \leftarrow 800$
 $PC \leftarrow 252$

(b) Salto condicional



Opcode: Bifurcación si $ACC=0$
Modo: Dirección directa
ADRS: 500
Operación: $PC \leftarrow 500$ si $ACC=0$
 $PC \leftarrow 302$ si $ACC \neq 0$

Tipos de direccionamiento

Indirecto

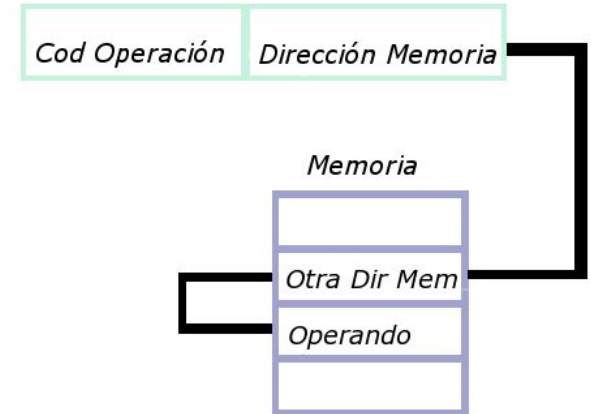
El campo de operando contiene una dirección de memoria, en la que se encuentra la dirección efectiva del operando.

Si hace referencia a un registro de la máquina, la dirección de memoria (dirección efectiva) que contiene el dato estará en este registro y hablaremos de direccionamiento indirecto a registro; si hace referencia a una posición de memoria, la dirección de memoria (dirección efectiva) que contiene el dato estará almacenada en esta posición de memoria y hablaremos de direccionamiento indirecto a memoria.

La desventaja principal de este modo de direccionamiento es que necesita un acceso más a memoria que el directo. Es decir, un acceso a memoria para el direccionamiento indirecto a registro y dos accesos a memoria para el direccionamiento indirecto a memoria; por este motivo este segundo modo de direccionamiento no se implementa en la mayoría de las máquinas. Ejemplo:

MOV A,@17H

Direccionamiento Indirecto



Tipos de direccionamiento

PC=250

ACC

Opcode: Carga indirecta ACC

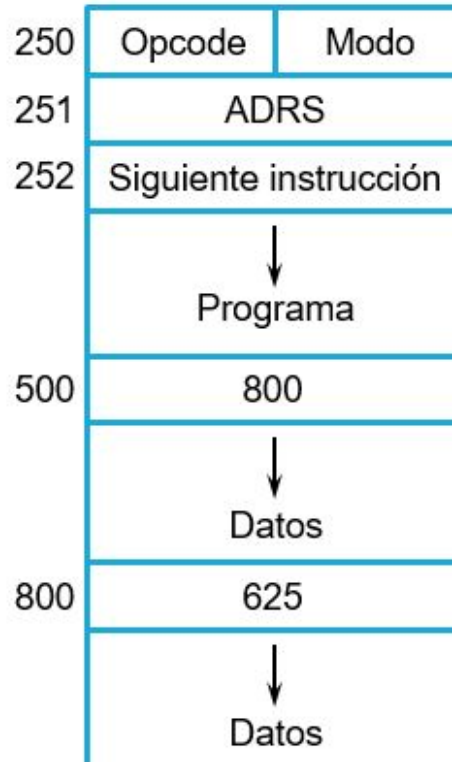
Modo: Dirección indirecta

ADRS: 500

Operación: $ACC \leftarrow 625$

$PC \leftarrow 252$

Operación: $ACC \leftarrow M[M[ADRS]]$



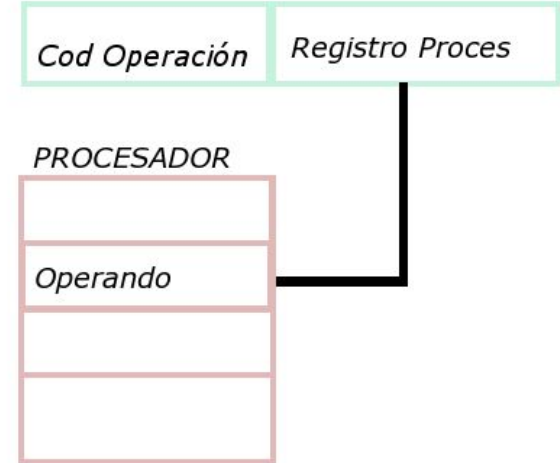
Tipos de direccionamiento

De registro

Sirve para especificar operandos que están en registros.

En este modo, los operandos están en registros que residen dentro de la CPU.

Direccionamiento por Registro



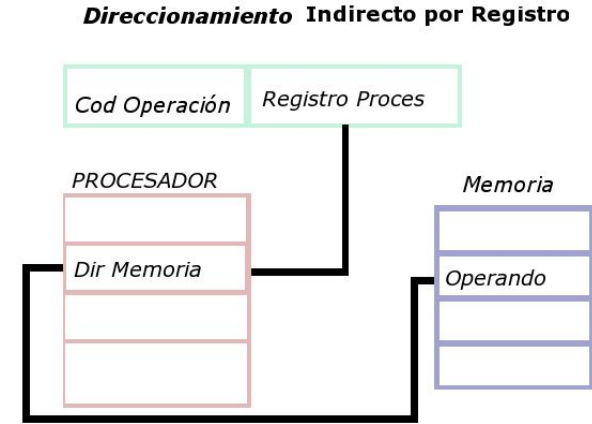
Tipos de direccionamiento

Indirecto mediante registros

El campo de operando de la instrucción contiene un identificador de registro en el que se encuentra la dirección efectiva del operando.

En este modo el campo de la dirección de la instrucción da la dirección en donde la dirección efectiva se almacena en la memoria. El control localiza la instrucción de la memoria y utiliza su parte de dirección para acceder a la memoria de nuevo para leer una dirección efectiva. Unos pocos modos de direccionamiento requieren que el campo de dirección de la instrucción sea sumado al control de un registro especificado en el procesador. La dirección efectiva en este modo se obtiene del siguiente cálculo:

Dir. efectiva = Dir. de la parte de la instrucción + Contenido del registro del procesador.

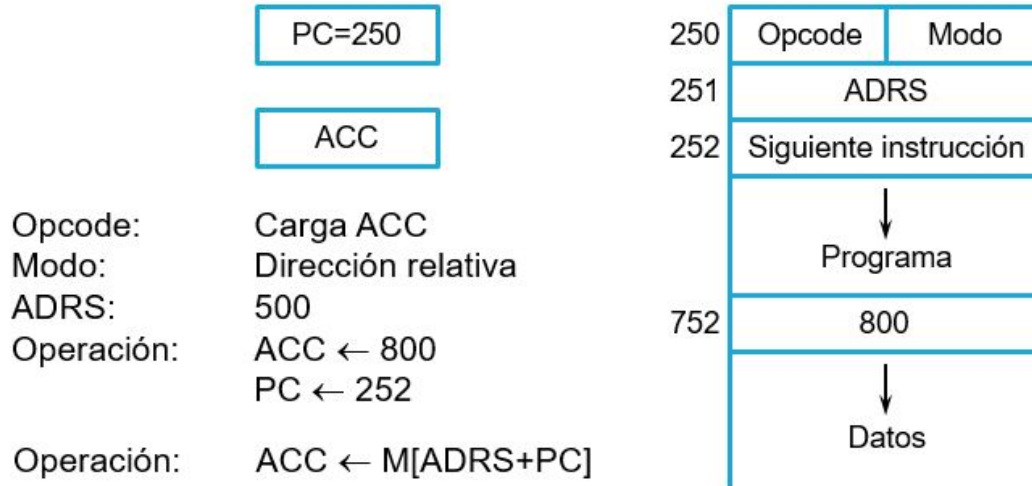


Tipos de direccionamiento

Relativo

La dirección efectiva se obtiene como suma de la parte de dirección de la instrucción y el contenido del PC. La parte de dirección de la instrucción es un número con signo.

Este tipo de direccionamiento se usa frecuentemente en instrucciones de salto condicional cuando la dirección de bifurcación está cercana a la instrucción de salto. Este direccionamiento genera instrucciones más compactas, ya que la dirección relativa necesita de menos bits que una dirección de memoria.




Tipos de direccionamiento

Relativo a un registro base

Consiste, al igual que el indirecto a través de registro, en calcular la dirección efectiva (EA, effective address) como la suma del contenido del registro base y un cierto desplazamiento (offset) que siempre será positivo. Esta técnica permite códigos reentrantes y acceder de forma fácil y rápida a posiciones cercanas de memoria. Este modo de direccionamiento es muy usado por los ensambladores cuando se llaman a las funciones (para acceder a los parámetros almacenados en la pila).

Relativo a un registro índice

Es similar al direccionamiento relativo a un registro base, excepto que es el contenido del registro índice el que indica el desplazamiento que se produce a partir de una dirección de memoria que se pasa también como argumento a la orden que utiliza este modo de direccionamiento. Aunque en esencia son dos modos equivalentes. La EA se calcula como la suma del contenido del registro índice y una dirección de memoria.



Tipos de direccionamiento

Modo de Registro BASE

La dirección efectiva se obtiene como suma de la parte de dirección de la instrucción y el contenido de un registro del procesador, llamado registro base.

Mientras que en el modo indexado el registro índice contiene un número que es relativo a la parte de dirección de la instrucción, en este modo el registro base es el que contiene la dirección base y el campo de direcciones de la instrucción proporciona el desplazamiento relativo.



Tipos de direccionamiento

INDEXADO

La dirección efectiva se obtiene como suma de la parte de dirección de la instrucción y el contenido de un registro del procesador, llamado registro índice. Este puede ser un registro especial del procesador o simplemente cualquiera de los registros del banco de registros.

Se usa cuando se desea acceder o manipular un array de datos almacenado en la memoria:

- El campo de direcciones de la instrucción marca el comienzo del array.
- Cada operando del array se almacena en memoria en una dirección relativa a su comienzo.
- La distancia entre la dirección de comienzo y la dirección del operando es el valor del índice almacenado en el registro.

Así, con la misma instrucción se puede acceder a cualquier operando del array mediante el uso del registro índice. Si los operandos están consecutivos en la memoria, se accede a cada uno de ellos incrementando el registro índice.



Tipos de direccionamiento

Indexado respecto a una base

Se trata de una combinación de los dos anteriores y consiste en calcular la dirección efectiva como:

Relativo al contador de programa

Consiste en dirección una posición de memoria usando como registro base al contador de programa (PC), el funcionamiento es análogo al direccionamiento respecto a registro base con la salvedad de que, en este caso, el offset puede ser también negativo.



Tipos de direccionamiento

RESUMEN

Para comparar los diferentes modos de direccionamiento, consideraremos como ejemplo una operación de carga del acumulador:

Opcode: Carga ACC
ADRS o NBR: 500

PC=250

R1=400

ACC

Modo	Nemónico	Transferencias	Dir. ef.	ACC
Directo	LDA ADRS	$ACC \leftarrow M[ADRS]$	500	800
Inmediato	LDA #NBR	$ACC \leftarrow NBR$	251	500
Indirecto	LDA [ADRS] LDA @ADRS	$ACC \leftarrow M[M[ADRS]]$	800	300
Relativo	LDA \$ADRS	$ACC \leftarrow M[ADRS+PC]$	752	600
Indexado	LDA ADRS (R1)	$ACC \leftarrow M[ADRS+R1]$	900	200
Registro	LDA R1	$ACC \leftarrow R1$	—	400
Registro indirecto	LDA (R1)	$ACC \leftarrow M[R1]$	400	700

250	Opcode	Modo
251	ADRS o NBR	
252	Siguiete instrucción	
400	700	
500	800	
752	600	
800	300	
900	200	

Direccionamiento de operandos

Por ejemplo en la instrucción ADD se necesita especificar tanto los datos que se van a sumar, como el destino del resultado. Esto se conoce como direccionamiento de operandos. Dichos operandos pueden encontrarse tanto en el banco de registros como en la memoria.


Si la dirección del operando forma parte de la instrucción, se tiene un direccionamiento explícito. En caso contrario, se tiene un direccionamiento implícito.

El número de operandos que se pueden direccionar explícitamente en una instrucción es un factor importante a tener en cuenta a la hora de definir la arquitectura del conjunto de instrucciones de un procesador.

Como ejemplo, supongamos que se quiere evaluar la siguiente expresión aritmética:

$$X = (A+B)(C+D)$$

con instrucciones de tres, dos, una y cero direcciones. Supondremos además que:

- Los operandos están en direcciones de memoria representadas por las letras A, B, C y D. La operación no debe alterar sus contenidos.
 - El resultado se almacena en la memoria, en una dirección representada por X.
 - Las operaciones aritméticas que se pueden utilizar son la suma ADD y la multiplicación MUL sobre dos operandos.
 - Las operaciones de transferencia de datos que se pueden utilizar son LD (de memoria a registro), ST (de registro a memoria) y MOVE (entre registros o posiciones de memoria).
- 

Tipos de instrucciones

INSTRUCCIONES DE TRES DIRECCIONES

Programa para el cálculo de $X = (A+B)(C+D)$ utilizando instrucciones de tres direcciones:

ADD T1, A, B	$M[T1] \leftarrow M[A] + M[B]$
ADD T2, C, D	$M[T2] \leftarrow M[C] + M[D]$
MUL X, T1, T2	$M[X] \leftarrow M[T1] \times M[T2]$

T1 y T2 son posiciones de memoria para almacenar los resultados parciales del cálculo.

ADD R1, A, B	$R1 \leftarrow M[A] + M[B]$
ADD R2, C, D	$R2 \leftarrow M[C] + M[D]$
MUL X, R1, R2	$M[X] \leftarrow R1 \times R2$

El mismo programa puede usar registros como posiciones de almacenamiento temporal:

El uso de registros puede reducir el tiempo de ejecución del programa entre 5 y 9 veces.

Ventaja: Se reducen las longitudes de los programas para la evaluación de expresiones.

Inconveniente: El código binario de la instrucción necesita más bits para poder especificar las tres direcciones (sobre todo si son posiciones de memoria).

Tipos de instrucciones

INSTRUCCIONES DE DOS DIRECCIONES

En la instrucción sólo hay dos direcciones explícitas (registros o posiciones de memoria). Se supone implícitamente que el resultado de la operación indicada en la instrucción se almacena en la dirección del primer operando indicado explícitamente.

Programa para el cálculo de $X = (A+B)(C+D)$ utilizando instrucciones de dos direcciones:

MOVE T1, A	$M[T1] \leftarrow M[A]$
ADD T1, B	$M[T1] \leftarrow M[T1] + M[B]$
MOVE X, C	$M[X] \leftarrow M[C]$
ADD X, D	$M[X] \leftarrow M[X] + M[D]$
MUL X, T1	$M[X] \leftarrow M[X] \times M[T1]$

Al igual que en el caso de las tres direcciones, se pueden usar registros en lugar de las posiciones de memoria para almacenar los cálculos parciales.

Para poder realizar el cálculo deseado hubo que:

- Introducir operaciones de transferencia de datos MOVE.
- Aumentar la longitud del programa.

Como contrapartida, la longitud de la instrucción es menor.



Tipos de instrucciones

INSTRUCCIONES DE UNA DIRECCIÓN

En la instrucción sólo hay una dirección explícita. El procesador utiliza implícitamente un registro acumulador ACC para poder realizar tanto las operaciones aritméticas, siendo la fuente de uno de los operandos y el destino del resultado, como las de transferencia de datos.

Programa para el cálculo de $X = (A+B)(C+D)$ utilizando instrucciones de una dirección:

LD	A	$ACC \leftarrow M[A]$
ADD	B	$ACC \leftarrow ACC + M[B]$
ST	X	$M[X] \leftarrow ACC$
LD	C	$ACC \leftarrow M[C]$
ADD	D	$ACC \leftarrow ACC + M[D]$
MUL	X	$ACC \leftarrow ACC \times M[X]$
ST	X	$M[X] \leftarrow ACC$

Ha aumentado el número de líneas de programa y el número de accesos a memoria.

Tipos de instrucciones

INSTRUCCIONES DE CERO DIRECCIONES

Para llevar a cabo operaciones aritméticas con cero direcciones, todas ellas deben ser implícitas. El procesador utiliza una estructura LIFO llamada pila (stack), de la que las instrucciones toman los datos y en donde almacenan los resultados:

- TOS (top of stack) es la palabra más arriba en la pila.
- TOS_{-1} es la palabra inmediatamente debajo, TOS_{-2} la siguiente, etc.
- Cuando se usan uno o varios operandos en una operación, se eliminan de la pila. La palabra por debajo pasa a ser la nueva TOS.
- Cuando se genera un resultado, se coloca en la pila pasando a ser el nuevo TOS.

De este modo, TOS y algunas palabras ubicadas por debajo son las direcciones implícitas de los operandos, y TOS es la dirección implícita del resultado. Por ejemplo, la instrucción de suma sería simplemente “ADD”, y la operación que se llevaría a cabo sería:

$$TOS \leftarrow TOS + TOS_{-1}.$$

Tipos de instrucciones

Para meter y sacar datos de la pila se usan las instrucciones:

- **PUSH (empujar).** PUSH X transfiere la palabra en la dirección X de la memoria al TOS.
- **POP (sacar).** POP X transfiere el TOS a la dirección X de la memoria.

Volviendo a nuestro cálculo de ejemplo, el programa con instrucciones de cero direcciones será el siguiente:

		<u>Pila</u>	<u>Notación posfijo</u>
PUSH A	$TOS \leftarrow M[A]$	<div>A</div>	A
PUSH B	$TOS \leftarrow M[B]$	<div>B A</div>	B
ADD	$TOS \leftarrow TOS + TOS_{-1}$	<div>A+B</div>	+
PUSH C	$TOS \leftarrow M[C]$	<div>C A+B</div>	C
PUSH D	$TOS \leftarrow M[D]$	<div>D C A+B</div>	D
ADD	$TOS \leftarrow TOS + TOS_{-1}$	<div>C+D A+B</div>	+
MUL	$TOS \leftarrow TOS \times TOS_{-1}$	<div>(A+B)(C+D)</div>	×
POP X	$M[X] \leftarrow TOS$		

Se necesitan en total ocho instrucciones, una más que en el caso de instrucciones de una dirección. Como contrapartida, sólo las instrucciones PUSH y POP utilizan direccionamiento explícito.

ARQUITECTURA DE DIRECCIONAMIENTO

Arquitectura memoria a memoria:

- Los operandos se toman directamente de la memoria, y el resultado se envía directamente a la memoria.
- Las instrucciones de manipulación y transferencia de datos contienen entre uno y tres campos de direcciones.
- El procesador sólo tiene registros de control.
- Es el caso del ejemplo del cálculo $X=(A+B)(C+D)$ con instrucciones de tres direcciones:

ADD T1, A, B

$M[T1] \leftarrow M[A] + M[B]$

ADD T2, C, D

$M[T2] \leftarrow M[C] + M[D]$

MUL X, T1, T2

$M[X] \leftarrow M[T1] \times M[T2]$

En la práctica este tipo de arquitecturas no se usa debido a que son lentas:

- Se necesita un elevado número de accesos a memoria, tanto en la fase de búsqueda de las instrucciones del programa como en la de su ejecución.
- Dar la posibilidad de que todas las instrucciones puedan acceder directamente a memoria incrementa la complejidad de las estructuras de control, dando como resultado que el período de la señal de reloj tenga que ser más largo.

ARQUITECTURA DE DIRECCIONAMIENTO

Arquitectura registro a registro y carga/almacenamiento:

- Permiten una sola dirección de memoria, y restringen su uso a las instrucciones de carga y almacenamiento.
- Las instrucciones de manipulación de datos acceden solamente a registros. - El procesador necesita un banco de registros con un tamaño adecuado.

En esta arquitectura, el programa ejemplo para el cálculo de $X=(A+B)(C+D)$ será:

LD	R1, A	$R1 \leftarrow M[A]$	PUSH A
LD	R2, B	$R2 \leftarrow M[B]$	PUSH B
ADD	R3, R1, R2	$R3 \leftarrow R1 + R2$	ADD
LD	R1, C	$R1 \leftarrow M[C]$	PUSH C
LD	R2, D	$R2 \leftarrow M[D]$	PUSH D
ADD	R1, R1, R2	$R1 \leftarrow R1 + R2$	ADD
MUL	R1, R1, R3	$R1 \leftarrow R1 \times R3$	MUL
ST	X, R1	$M[X] \leftarrow R1$	POP X

Este tipo de arquitecturas se usa más en la práctica porque, aunque se necesiten más instrucciones en el programa, el número total de accesos a memoria se reduce y, por tanto, son más rápidas.

Tipos de Instrucciones

Instrucciones de transferencia de datos.

Las instrucciones de transferencia de datos mueven un dato de un lugar del procesador a otro, sin modificarlo.

Las típicas transferencias son entre memoria y los registros del procesador, entre estos y los registros de entrada/salida (E/S), y entre los propios registros de procesador:

Nombre	Nemónico	Descripción
Load	LD	Desde una posición de memoria a un registro del procesador.
Store	ST	Desde un registro del procesador a una posición de memoria.
Move	MOVE	Desde un registro del procesador a otro. También se utiliza para transferir datos entre dos posiciones de memoria.
Exchange	XCH	Intercambia datos entre dos registros del procesador o entre dos posiciones de memoria.
Push	PUSH	Transferencia de datos con la pila
Pop	POP	
Input	IN	Transferencia de datos con registros E/S
Output	OUT	

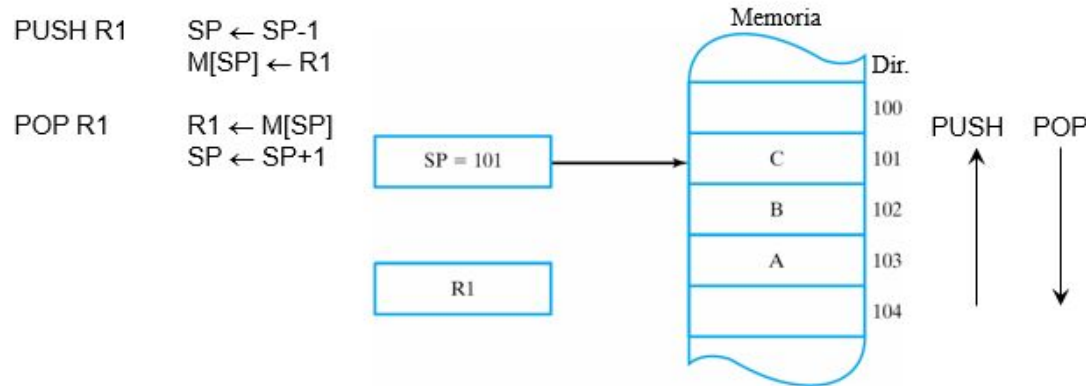
Tipos de Instrucciones

TRANSFERENCIA DE DATOS CON LA PILA

La arquitectura basada en pila posee características que facilitan ciertos tipos de procesamiento de datos y de control de tareas. Así, en algunas calculadoras y procesadores se usan pilas para la evaluación de expresiones aritméticas.

Las pilas en memoria se organizan mediante un registro especial, llamado puntero de pila SP (Stack Pointer). El registro SP contiene la dirección del TOS.

PUSH es la instrucción que almacena datos en la pila, y POP la que los extrae de ella:



SP se inicializa con la dirección de memoria donde quiera iniciarse la pila (104 en el ejemplo).

Tipos de Instrucciones - Instrucciones de manipulación de datos

Las instrucciones de manipulación de datos realizan operaciones sobre los datos, proporcionando los recursos de cálculo del procesador. Son de tres tipos:

1. Instrucciones aritméticas.
2. Instrucciones lógicas y de manipulación de bits.
3. Instrucciones de desplazamiento.



Tipos de Instrucciones - Instrucciones de manipulación de datos

INSTRUCCIONES ARITMÉTICAS

Nombre	Nemónico	Descripción
Incremento	INC	Suma uno. Si todos los bits son 1, genera una palabra con todos los bits a 0.
Decremento	DEC	Restar uno. Si todos los bits son 0, genera una palabra con todos los bits a 1.
Suma	ADD	En procesadores sencillos sólo están disponibles las operaciones de suma y de resta. En esos casos, la multiplicación y la división se tienen que hacer mediante programas. Normalmente estas operaciones están disponibles para diferentes tipos de datos.
Resta	SUB	
Multiplicación	MUL	
División	DIV	
Suma con acarreo	ADDC	Suma dos <u>operandos</u> teniendo en cuenta el acarreo del cálculo anterior. Permite cálculos de doble precisión.
Resta con acarreo	SUBB	Resta dos <u>operandos</u> teniendo en cuenta el acarreo (<u>borrow</u>) del cálculo anterior. Permite cálculos de doble precisión.
Resta inversa	SUBR	Invierte el orden de los <u>operandos</u> , haciendo la resta B-A en lugar de A-B.
Negativo	NEG	Halla el complemento a dos del operando.

Tipos de Instrucciones - Instrucciones de manipulación de datos




INSTRUCCIONES LÓGICAS Y DE MANIPULACIÓN DE BITS

Nombre	Nemónico	Descripción
Poner a 0 (Clear)	CLR	Pone todos los bits del operando a 0.
Poner a 1 (Set)	SET	Pone todos los bits del operando a 1.
Complementar	NOT	Complementa los bits del operando.
Producto lógico	AND	Realizan la función lógica correspondiente entre dos <u>operandos</u> bit a bit.
Suma lógica	OR	AND se usa con máscaras para poner selectivamente a 0 algunos bits
OR exclusiva	XOR	de los <u>operandos</u> , OR para ponerlos a 1, y XOR para complementarlos.
Poner a 0 el acarreo (Clear <u>Carry</u>)	CLRC	Estas instrucciones sobre el bit de acarreo pueden definirse para los demás bits de estado del procesador.
Poner a 1 el acarreo (Set <u>Carry</u>)	SETC	
Complementar el acarreo	COMC	

Tipos de Instrucciones - Instrucciones de manipulación de datos

INSTRUCCIONES DE DESPLAZAMIENTO

Permiten desplazar o rotar los bits de un operando de diversas formas:

Nombre	Nemónico	Descripción
Desplazamiento lógico a la derecha	SHR	
Desplazamiento lógico a la izquierda	SHL	
Desplazamiento aritmético a la derecha	SHRA	
Desplazamiento aritmético a la izquierda	SHLA	
Rotación a la derecha	ROR	
Rotación a la izquierda	ROL	
Rotación a la derecha con acarreo	RORC	
Rotación a la izquierda con acarreo	ROLC	

Algunos procesadores tienen un formato para la instrucción de desplazamiento con varios campos:

- 1º) OP: Código de operación que indica un desplazamiento.
- 2º) REG: Dirección que especifica la localización del operando.
- 3º) TYPE: Código de dos bits que indica el tipo de desplazamiento.
- 4º) RL: Código de un bit que indica la dirección de desplazamiento.
- 5º) COUNT: Código de k bits que indica el número de posiciones a desplazar.

Tipos de Instrucciones - Instrucciones de control de programa

INSTRUCCIONES DE SALTO CONDICIONAL

Comprueban los bits de estado para una determinada condición: si es cierta, el control de programa se transfiere a la dirección efectiva, y si es falsa se continúa con la ejecución de la instrucción siguiente.

Condición de bifurcación	Nemónico	Test	Descripción
Bifurcación si es cero	BZ	$Z = 1$	El bit de estado Z se usa para comprobar si el resultado de una operación de la ALU es 0.
Bifurcación si no es cero	BNZ	$Z = 0$	
Bifurcación si hay acarreo	BC	$C = 1$	El bit de acarreo C se usa para comprobar el acarreo después de una suma o resta en la ALU. También se utiliza en las instrucciones de desplazamiento para comprobar el bit saliente.
Bifurcación si no hay acarreo	BNC	$C = 0$	
Bifurcación si negativo	BN	$N = 1$	El bit de estado N refleja el estado del bit más significativo de la salida de la ALU, tanto si representa el signo como si no.
Bifurcación si positivo	BNN	$N = 0$	
Bifurcación si hay desbordamiento	BV	$V = 1$	El bit de desbordamiento V se usa en operaciones aritméticas de números con signo.
Bifurcación si no hay desbordamiento	BNV	$V = 0$	

Tipos de Instrucciones - Instrucciones de control de programa

INSTRUCCIONES DE SALTO CONDICIONAL

Las comparaciones de dos palabras A y B se hacen mediante una operación de resta (A-B). El resultado no se almacena, pero los bits de estado se ven afectados.

Para números binarios sin signo:

Condición de bifurcación	Nemónico	Condición	Bits de estado
Bifurcación si es mayor	BH	$A > B$	$C+Z = 0$
Bifurcación si es mayor o igual	BHE	$A \geq B$	$C = 0$
Bifurcación si es menor	BL	$A < B$	$C = 1$
Bifurcación si es menor o igual	BLE	$A \leq B$	$C+Z = 1$
Bifurcación si es igual	BE	$A = B$	$Z = 1$
Bifurcación si no es igual	BNE	$A \neq B$	$Z = 0$

Para números binarios con signo:

Condición de bifurcación	Nemónico	Condición	Bits de estado
Bifurcación si es mayor	BG	$A > B$	$(N \oplus V)+Z = 0$
Bifurcación si es mayor o igual	BGE	$A \geq B$	$N \oplus V = 0$
Bifurcación si es menor	BL	$A < B$	$N \oplus V = 1$
Bifurcación si es menor o igual	BLE	$A \leq B$	$(N \oplus V)+Z = 1$
Bifurcación si es igual	BE	$A = B$	$Z = 1$
Bifurcación si no es igual	BNE	$A \neq B$	$Z = 0$

Tipos de Instrucciones - Subrutinas

SUBROUTINAS

Es relativamente frecuente que conjuntos de instrucciones en un programa se repitan. En lugar de copiar varias veces en memoria las mismas secuencias de instrucciones, se almacenan una única vez en lo que se denomina subrutina. Cada vez que se necesite ejecutar esa secuencia se hace un salto a la primera instrucción de la subrutina. Cuando se termine de ejecutar la subrutina, se hace otro salto para volver al programa principal.

La instrucción de llamada a subrutina CALL posee un campo de dirección y realiza dos operaciones:

1. Guarda el valor del PC en una zona de almacenamiento temporal (dirección de retorno).
2. La dirección de la instrucción CALL se carga en el PC (primera instrucción de la subrutina).

La última instrucción de una subrutina es la de retorno al programa principal RET: carga en el PC la dirección almacenada por la instrucción CALL.

Algunos procesadores utilizan una pila como almacenamiento temporal de las direcciones de retorno, lo cual permite manejar subrutinas anidadas:



Tipos de Instrucciones - Subrutinas

CALL dir:

$SP \leftarrow SP - 1$

$M[SP] \leftarrow PC$

$PC \leftarrow dir$

Decrementa el puntero de pila.

Almacena la dirección de retorno en la pila.

Transfiere el control a la subrutina.

RET:

$PC \leftarrow M[SP]$

$SP \leftarrow SP + 1$

Transfiere la dirección de retorno al PC.

Incrementa el puntero de pila.



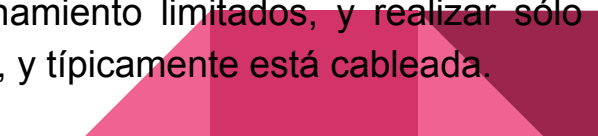
Arquitecturas de conjunto de instrucciones

ARQUITECTURA RISC

Los procesadores de conjunto reducido de instrucciones RISC (Reduced Instruction Set Computers) tienen las siguientes propiedades:

1. Las instrucciones sólo realizan operaciones elementales.
2. Los accesos a memoria se restringen a las instrucciones de carga y almacenamiento. Las instrucciones de manipulación de datos son de registro a registro.
3. Número limitado de modos de direccionamiento.
4. Los formatos de todas las instrucciones tienen la misma longitud.

Características de la arquitectura RISC:

1. Su objetivo es conseguir un alto rendimiento (alta velocidad de ejecución). Por ello los accesos a memoria se limitan a las instrucciones de carga y almacenamiento.
 2. Necesita un banco de registros relativamente grande.
 3. Al tener las instrucciones una longitud fija, modos de direccionamiento limitados, y realizar sólo operaciones básicas, la unidad de control es relativamente simple, y típicamente está cableada.
 4. La organización interna del procesador es en pipeline.
- 

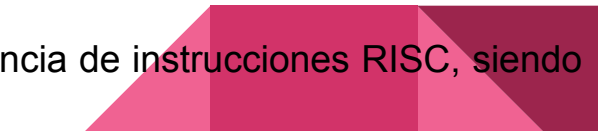
Arquitecturas de conjunto de instrucciones

ARQUITECTURA CISC

Los procesadores de conjunto de instrucciones complejo CISC (Complex Instruction Set Computers) tienen las siguientes propiedades:

1. Las instrucciones realizan tanto operaciones elementales como complejas.
2. Los accesos a memoria están disponibles en prácticamente todos los tipos de instrucciones.
3. Gran número de modos de direccionamiento.
4. Los formatos de las instrucciones son de diferente longitud.

Características de la arquitectura CISC:

1. Su objetivo es proporcionar instrucciones que faciliten realizar programas compactos y así ahorrar memoria.
 2. Debido a la alta accesibilidad de la memoria, el banco de registros es reducido.
 3. Debido a la complejidad de las instrucciones y la diversidad de sus formatos, la unidad de control es compleja y suele utilizar microprogramación.
 4. Internamente, las instrucciones CISC se convierten en una secuencia de instrucciones RISC, siendo procesadas mediante un pipeline tipo RISC.
- 

Diferencias

	RISC	CISC
1	Instrucciones sencillas en un ciclo	Instrucciones complejas en varios ciclos
2	Sólo LOAD/STORE hacen referencia a memoria	Cualquier instrucción puede referencia memoria
3	Procesamiento serie de varias etapas	Poco procesamiento en serie
4	Instrucciones ejecutadas por hardware	Instrucciones interpretadas por el microprograma
5	Instrucciones de formato fijo	Instrucciones de formato variable
6	Pocas instrucciones y modos de direccionamiento	Muchas instrucciones y modos de direccionamiento
7	La complejidad está en el compilador	La complejidad está en el microprograma
8	Varios conjuntos de registros	Un sólo conjunto de registros
9	Mayor uso del Bus en Búsqueda	Menor uso del Bus en Búsqueda
10	Menor uso del Bus en Ejecución	Mayor uso del Bus en Ejecución